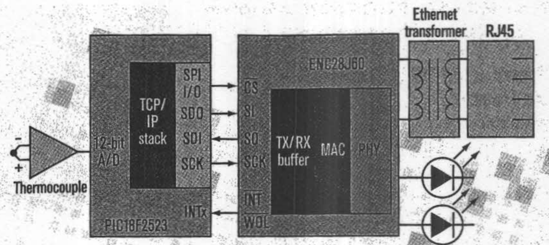
STEP-BY-STEP DESIGN

Adding Network Capability to your Embedded Application

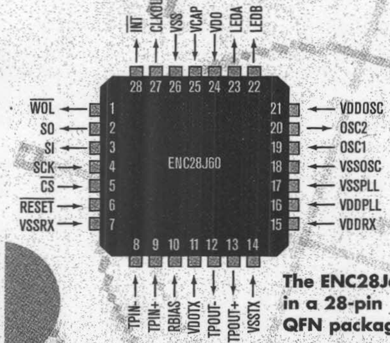
Rodger Richey,
Applications Manager,
Advanced Microcontroller Architecture Division,
Microchip Technology Inc.



INTRODUCTION

With the increased horsepower of 8-bit microcontrollers (MCUs), the integration of an embedded TCP/IP stack is quite feasible. For many years now, TCP/IP stacks have been available from MCU manufacturers and their third-party tool providers. These stacks were developed specifically for 8-bit MCUs, which have program memory and RAM. Due to optimization, 8-bit MCU implementations are quite capable. Figure 1 shows a very simple sensor application where an 8-bit MCU with a 12-bit a/d converter is connected to a 10 BASE-T Ethernet interface controller and is used to display the sensor information.

One misconception is that a 10 BASE-T device, when placed in a network with 100 BASE-T devices, will drag the network down to 10 Mb/s. Most switches and routers in use today will convert the 10 Mb/s stream on one channel into 100 Mb/s or faster on the uplink–thus there is no slowdown. Let's get started with the design.



The ENC28J60 in a 28-pin QFN package.

1

SELECT THE ETHERNET INTERFACE

There are many Ethernet interface devices available, but most are aimed at high-speed parallel buses. These devices can be very cumbersome for 8-bit MCUs, since they consume numerous I/O pins. This forces designers to use higher pin-count devices, which thereby decreases the performance of the TCP/IP stack, since the interface is manipulated manually rather than through a standard peripheral on the 8-bit MCU. The ENC28J60 Ethernet Interface chip from Microchip Technology (see Figure 2) was selected for this application because of its small size (28-pin QFN), integrated 8 KB of RAM and its SPI interface.

2

SELECT THE MICROCONTROLLER

Selection of the MCU comes down to memory requirements and peripheral set. A typical TCP/IP stack uses approximately 20 KB of program memory. This means that an MCU with about 32 KB should be used, allowing 12 KB for the application. For our networked sensor, 12 KB is more than enough room for the main application code. A microcontroller such as the PIC18F2523 from Microchip Technology provides a 10-channel, 12-bit a/d converter, dual analog comparators, PWM, SPI, I^{2}C, and USART peripherals

3

SELECT THE MEMORY

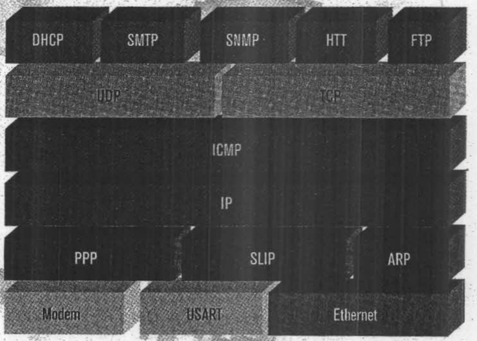
One consideration when creating an embedded Web server is where the Web pages will be stored. Very simple designs with one or two pages can be stored in the MCU's internal program memory. However, marketing departments will most likely want something more than that. Several options exist for external memory, including serial EEPROMs and flash memory. Serial EEPROMs can provide up to 512 Kb of storage space, while serial flash devices offer up to 32 Mb.

4

SELECT THE PROTOCOLS

An important step in designing an embedded TCP/IP application is the selection of protocols to transfer the information. TCP provides a robust handshaking protocol to reliably transfer data. It is also over three times the size of the user datagram protocol (UDP). If the application must serve Web pages, then TCP is required. However, if the application only needs to communicate the information to a collection point, firmware can be added to the UDP to make it more robust and still be far smaller than TCP.

Other protocols needed are DHCP to dynamically configure IP addresses, ARP to resolve addresses, and IP. ICMP may also be useful for the ping command. This allows the application to respond to echo requests, or pings, which help to debug the connection of devices on the network. Figure 4 shows the typical protocol stack for TCP/IP.



Typical TCP/IP Protocol Stack

5

CREATE THE WEB PAGES

The final part of the design is to create Web pages. Web-page creation software adds a lot of extra information into HTML Web pages, which takes up space. Do you really want to fill up your precious serial EEPROM with comments? The best tools to create HTML code may be Windows Notepad and a good book on HTML.

Keep in mind that although graphics consume a lot of memory, they provide a rich user interface. An innovative solution to limit the amount of memory required for this project is to store the graphics on a server on the network, then have the Web page reference the graphics.

CONCLUSION

Once the hardware is complete and the Web pages are done, you can download the pages to memory and power up the sensor. Then it's just a matter of connecting the device to the network and typing in the IP address on a browser to see the sensor information displayed in all its glory.